AMENDMENTS TO THE CLAIMS

The following listing of claims will replace all prior versions and listings of claims in the application.

LISTING OF CLAIMS

1. (Previously Presented) A method used while building in processor memory a stack of device objects (DOs) representing a device, there being a multi-role driver for a plurality of roles at least one of which corresponds to the device, the method comprising:

registering a plurality of uni-role helper divers so as to uniquely correspond to the plurality of roles, respectively,

each helper driver mapping uniquely to one of the multiple roles of the multi-role driver, respectively; and

indirectly specifying a corresponding one of the multiple roles of the multi-role driver by specifying the helper driver mapped thereto.

2. (Original) The method of claim 1, wherein the multi-role driver and the helper drivers are operable to run in the WINDOWS Driver Model environment.

3. (Original) The method of claim 1, wherein a role is determined according to a device type for which the multi-role driver is invoked and the extent of the stack at the point at which the multi-role driver is invoked.

4. (Original) The method of claim 1, wherein each of the multiple roles in the multi-role driver has a corresponding DOPush function, each DOPush function having been made available to be invoked by a code portion external to the multi-role driver.

5. (Original) The method of claim 1, wherein each helper driver includes an AddDevice routine that invokes a corresponding DOPush function in the multi-role driver, or each helper driver points to the address of the corresponding DOPush function in the multi-role driver.

6. (Previously Presented) A method used while assembling in processor memory a stack of device objects (DOs) representing a device, the operating system of the processor having a kernel, the device having a corresponding physical device object (PDO), the method comprising:

determining a uni-role first driver registered to the device;

invoking the first driver, which includes passing the PDO of the device to the first driver; and

passing the PDO from the first driver to the multi-role second driver or to a component of the kernel.

7. (Original) A method used while assembling in processor memory a stack of device objects (DOs) representing a device, the device having a corresponding physical device object (PDO), the method comprising:

determining a driver registered to the device;

invoking the driver, which includes passing the PDO of the device to the driver; and

passing the PDO away from the driver without attempting to attach to the stack a DO corresponding to the driver.

8. (Original) A method used while assembling in processor memory a stack of device objects (DOs) representing a device, there being a multi-role driver for a plurality of roles at least one of which corresponds to the device, the device having a corresponding physical device object (PDO), the method comprising:

providing a plurality of DOPush functions in a multi-role driver;

loading the multi-role driver into the memory so as to arrange for each of the DOPush functions to be directly invokable by a code portion external to the multi-role driver; and

invoking, externally to the multi-role driver, one of the DOPush functions, which includes passing the PDO of the device to the invoked DOPush function.

9. (Original) The method of claim 8, wherein the DOPush function is invoked externally by

an AddDevice routine of a helper driver, or

the PnP manager, if the helper driver does not have the AddDevice routine, after the PnP manager is pointed to the address of the DOPush function by the helper driver,

the helper driver being registered uniquely for the role to which the DOPush function corresponds.

10. (Original) The method of claim 8, wherein the multi-role driver is operable to run in the WINDOWS Driver Model environment.

11. (Original) The method of claim 8, further comprising: registering neither the multi-role driver nor the DOPush functions in the registry of the operating system as having a role in assembly of a stack representing a device.

12. (Original) The method of claim 8, wherein a role is determined according to a device type for which the multi-role driver is invoked and the extent of the stack at the point at which the multi-role driver is invoked.

13. (Original) A method used while assembling in processor memory a stack of device objects (DOs) representing a device, the method comprising:

providing a multi-role driver for a plurality of device types; but

not registering, in the registry of the operating system, the multi-role driver as having a role in assembly of the stack.

14. (Original) The method of claim 13, wherein the multi-role driver is operable to run in the WINDOWS Driver Model environment.

15. (Original) A code arrangement on a machine-readable medium execution of which facilitates assembling in processor memory a stack of device objects (DOs) representing a device, the machine-readable code arrangement comprising:

a multi-role driver code portion which corresponds to the device, the multi-role driver code portion having exported functions corresponding to the multiple roles of the multi-role driver code portion, respectively;

a plurality of helper driver code portions; and

an installer code portion for registering the plurality of helper driver code portions so as to uniquely map to the multiple roles, respectively;

each helper driver code portion being operable to receive a corresponding PDO and pass the PDO to the multi-role driver code portion without attempting to attach to the stack a DO corresponding to the helper driver code portion.

16. (Original) The machine-readable code arrangement of claim 15, wherein the multi-driver code portion and the helper driver code portions are operable to run in the WINDOWS Driver Model environment.

17. (Original) The machine-readable code arrangement of claim 15, wherein a role is determined according to a device type for which the multi-role driver code portion is invoked and the extent of the stack at the point at which the multi-role driver code portion is invoked.

18. (Original) The machine-readable code arrangement of claim 15, wherein the exported functions are DOPush functions.

19. (Original) The machine-readable code arrangement of claim 18, wherein each helper driver code portion includes an AddDevice routine code portion that invokes the corresponding DOPush function in the multi-role driver code portion, or each helper driver code portion is operable to point to the address of the corresponding DOPush function in the multi-role driver code portion.

20. (Original) An apparatus having memory in which is buildable a stack of device objects (DOs) representing a device attached to the apparatus, the apparatus comprising:

multi-role driver means for operating according to a plurality of roles;

a plurality of helper driver means registered so as to uniquely correspond to the plurality of roles, respectively, of the multi-role driver; and

means for selectively invoking the multi-role driver according to one of the multiple roles via invoking the corresponding helper driver mapped thereto.

21. (Original) The apparatus of claim 20, wherein the multi-role driver means and the helper driver means are operable to run in the WINDOWS Driver Model environment.

22. (Original) The apparatus of claim 20, wherein a role is determined according to a device type for which the multi-role driver means is invoked and the extent of the stack at the point at which the multi-role driver means is invoked.

23. (Original) The apparatus of claim 20, wherein the multi-role driver means provides a plurality of DOPush functions corresponding to the multiple roles, respectively, each DOPush function having been made available to be invoked by a code portion external to the multi-role driver means.

24. (Original) A code arrangement on a machine-readable medium execution of which facilitates building in processor memory a stack of device objects (DOs) representing a device, there being a multi-role driver for a plurality of roles at least one of which corresponds to the device, the machine-readable code arrangement comprising:

a plurality of helper driver code portions;

a first code portion for registering the plurality of helper driver code portions so as to uniquely correspond to the plurality of roles, respectively, each helper driver code portion mapping uniquely to one of the multiple roles of the multi-role driver, respectively; and

a second code portion for indirectly specifying a corresponding one of the multiple roles of the multi-role driver by specifying the helper driver code portion mapped thereto.

25. (Original) The machine-readable code arrangement of claim 24, wherein the multi-role driver and the helper driver code portions are operable to run in the WINDOWS Driver Model environment.

26. (Original) The machine-readable code arrangement of claim 24, wherein a role is determined according to a device type for which the multi-role driver is invoked and the extent of the stack at the point at which the multi-role driver is invoked.

27. (Original) The machine-readable code arrangement of claim 24, wherein each of the multiple roles in the multi-role driver has a corresponding DOPush function, each DOPush function having been made available to be invoked by a code portion external to the multi-role driver.

28. (Original) The machine-readable code arrangement of claim 24, wherein each helper driver code portion includes an AddDevice routine code portion that invokes a corresponding DOPush function in the multi-role driver, or each helper driver code portion points to the address of the corresponding DOPush function in the multi-role driver.